Rapid Report

ibc Interdisciplinary Bio Central

Open Access, Open Review Journal

Optimized Entity Attribute Value Model: A Search Efficient Representation of High Dimensional and Sparse Data

Razan Paul[1] and Abu Sayed Md. Latiful Hoque[1,*]

[1]Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh

Subject areas:
Biological computation/Database

***Correspondence** and requests for materials should be addressed to A.S.M.L.H. (asmlatifulhoque@cse.buet.ac.bd)

Editor: Hong Gil Nam, POSTECH, Republic of Korea

Received June 30, 2011;
Accepted July 06, 2011;
Published July 06, 2011

Citation: Paul, R., et al. Optimized Entity Attribute Value Model: A Search Efficient Representation of High Dimensional and Sparse Data. IBC 2011, 3:9, 1-5.
doi: 10.4051/ibc.2011.3.3.0009

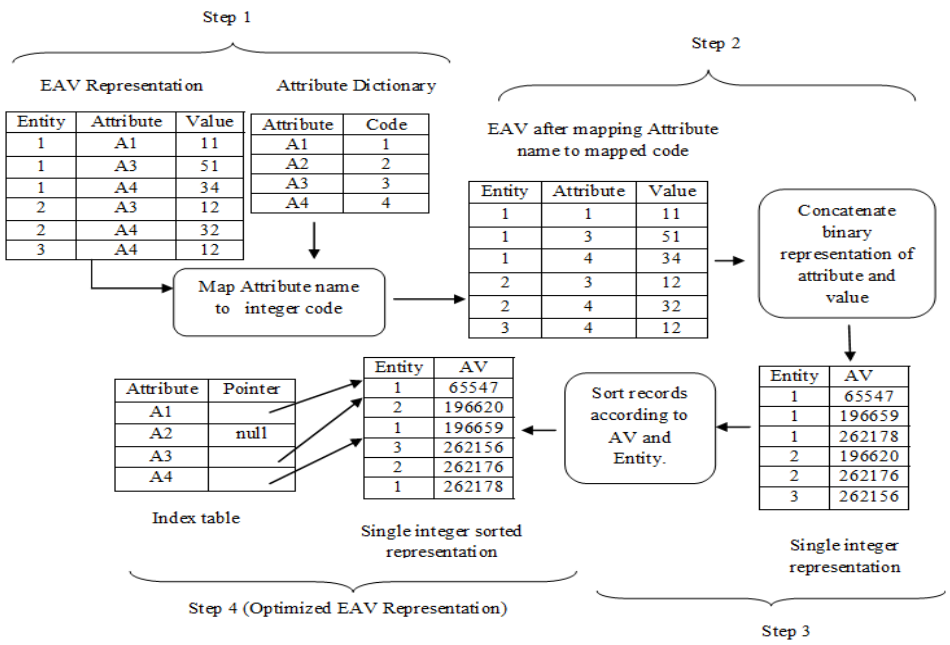
Funding: Research funding of BUET

Competing interest: All authors declare no financial or personal conflict that could inappropriately bias their experiments or writing.

This article is part of the special issue: the 9th Asia Pacific Bioinformatics Conference (APBC2011).

SYNOPSIS

Entity Attribute Value (EAV) is the widely used solution to represent high dimensional and sparse data, but EAV is not search efficient for knowledge extraction. In this paper, we have proposed a search efficient data model: Optimized Entity Attribute Value (OEAV) for physical representation of high dimensional and sparse data as an alternative of widely used EAV. We have implemented both EAV and OEAV models in a data warehousing environment and performed different relational and warehouse queries on both the models. The experimental results show that OEAV is dramatically search efficient and occupy less storage space compared to EAV.



Keywords: EAV, OEAV, open schema, sparse data, high dimensional data

Introduction

In various domains, integrating Large-scale heterogeneous data is important for knowledge discovery. Once this knowledge is discovered, the results of such analysis can be used to define new guidelines. We require an open schema data model to support dynamic schema change, sparse data, and high dimensional data. In open schema data models, logical model of data is stored as data rather than as schema, so changes to the logical model can be made without changing the schema. In open schema data model, schema is kept as data. EAV[1] is the widely used open schema data model to handle these challenges of data representation. However, EAV suffers from higher storage requirement and not search efficient. In this paper, we have proposed a search efficient open schema data model: OEAV. This model is storage efficient as well compared to existing EAV model.

Section 2 describes the related work. The overview of EAV, the details organizational structure and analysis of OEAV are given in section 3. A data transformation is required to adopt the existing data suitable for data warehouse representation for knowledge extraction. The transformation is elaborated in section 4. Analytical details of performance of the proposed models are given in section 5. Section 6 offers the result and discussion. Section 7 is the conclusion.

Related Work

EAV gives us extreme flexibility in data representation but it is not search efficient as it keeps attribute name as data in attribute column and has no tracking of how data are stored. To handle the high dimensionality and sparseness of medical data, in Thomas et al.[2] authors have used EAV, but EAV is not a search efficient data model for knowledge discovery. To handle data sparseness and schema change of phenotype data, the EAV model has been used for phenotype data management in Li et al.[3]. Use of EAV for medical observation data is also found in Anhoj et al.[4], Brandt et al.[5], Nadkarni et al.[6], Nadkarni, P.M.[7] as medical observation data are sparse, high dimensional and need frequent schema change. The Entity-Relationship Model is proposed in Pin-Shan Peter, C.[8]. Storage and querying of high dimensional sparsely populated data is proposed in Latiful Hoque, A.S.M.[9]. But this model does not keep the data in search efficient way. Agarwal et al.[10] propose several methods for the efficient computation of multidimensional aggregates. In Adam et al.[11] authors propose data cube as a relational aggregation operator generalizing group-by, crosstab, and subtotals.

Open Schema Data Models

A. Entity-Attribute-Value Model (EAV)

EAV is an open schema data model, which is suitable for high dimensional and sparse data like medical data. In EAV, every fact is conceptually stored in a table, with three sets of columns: entity, an attribute, and a value for that attribute. In this design, one row actually stores a single fact. It eliminates sparse data to reduce database size and allows changing set of attributes. Moreover, EAV can represent high dimensional data, which cannot be modeled by relational model because existing RDBMS only support a limited number of columns. EAV gives us extreme flexibility but it is not search efficient as it keeps attribute name as data in attribute column and has no tracking of how data are stored.

B. Optimized Entity Attribute Value (OEAV)

To remove the search inefficiency problem of EAV whilst preserving its efficiency of representing high dimensional and sparse data, we have developed a search efficient open schema data model OEAV. This model keeps data in a search efficient way.

This approach is a read-optimized representation whereas the EAV approach is write-optimized. Most of the data warehouse systems write once and read many times, so the proposed approach can serve the practical requirement of data warehouse. Figure 1 shows the step by step approach of transformation of an EAV data representation to an equivalent OEAV data representation. In step 1,

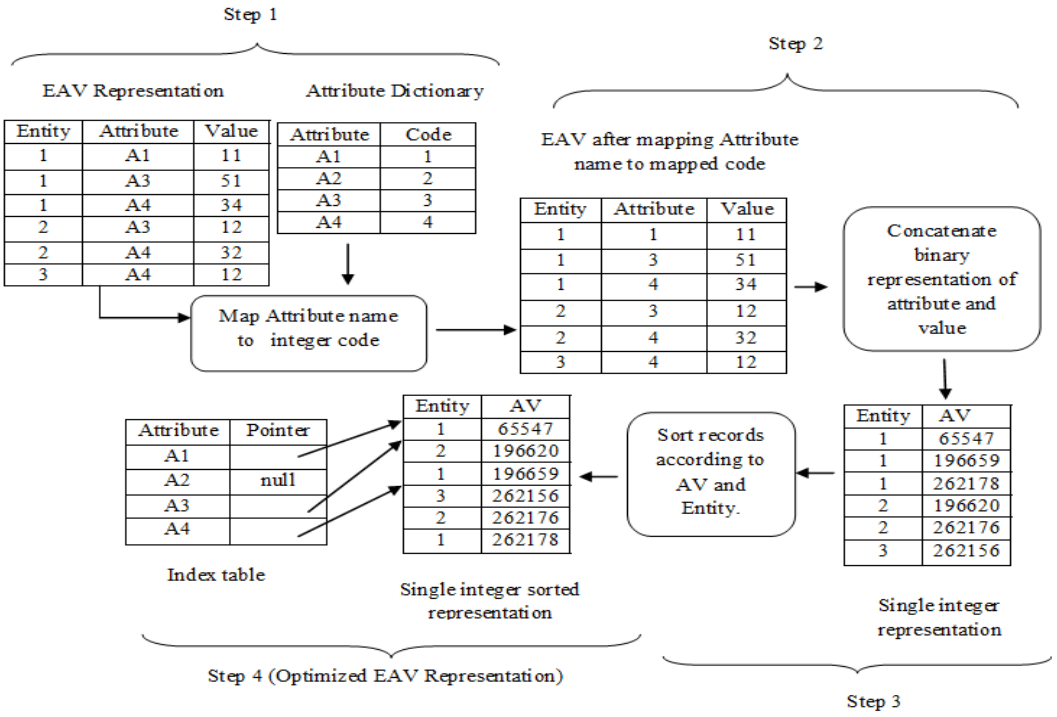


Figure 1. Transformation of EAV model to Optimized EAV (OEAV) model.

this model constructs an attribute dictionary where there is an integer code for each attribute.

Attribute name of each fact is mapped to an integer code using the attribute dictionary. All types of values are treated as integer using a data transformation as discussed in the following section. In step2, a compact single integer Attribute Value (AV) is created by concatenating binary representation of attribute code and value. In OEAV, every fact is conceptually stored in a table with two columns: the entity and the AV. It maps attribute code and value to p bit and q bit integer and concatenate them to n bit integer AV. For example, an attribute value pair (A3, 51), the code of attribute A3 is 3, will be converted in the following ways: (A3, 51) → (3, 51) → (0000000000 000011, 0000000000110011) → 00000000000000011000000000001 10011 = 196659.

In step 3, the records of optimized EAV are stored as sorted order of AV field. As data are stored in sorted order of AV and the first p bits of AV are for attribute code, the records of an attribute in OEAV table remains consecutively. In step 4, an index structure, which is a part of OEAV representation, is created to contain the starting record number (Pointer) of each attribute in OEAV table. This makes the data partitioned attribute wise, which is expected by most analytical program. In sorted AV field, the values of an attribute also remain in sorted order and binary search can be applied on it. This model constructs a modified B+ tree index on entity field of OEAV to make entity wise search efficient. Here each leaf of the modified B+ tree keeps the block address of attribute values for each entity. These Search efficiencies of OEAV are absent in conventional EAV representation.

Data Transformation Using Domain Dictionary and Rule Base

For knowledge discovery, the data have to be transformed into a suitable transaction format to discover knowledge. We address the problem of mapping data to items using domain dictionary and rule base. The data are type of categorical, continuous numerical data, Boolean, interval, percentage, fraction and ratio. Domain expert have the knowledge how to map ranges of numerical data for each attribute to a series of items. For example, there are certain conventions to consider a person is young, adult, or elder with respect to age. A set of rules is created for each continuous numerical attribute using the knowledge of domain experts. A rule engine is used to map continuous numerical data to items using these developed rules.

Data, for which domain expert knowledge is not applicable; we have used domain dictionary approach to transform these data to numerical forms. Here the mapping process as shown in Figure 2 for medical data is divided in two phases. Phase 1: a rule base is constructed based on the knowledge of domain experts and dictionaries are constructed for attributes where domain expert know ledge is not applicable, Phase 2: attribute values are mapped to integer values using the corresponding rule base and the dictionaries.

Storage Analysis of EAV & OEAV

Let b be the total number of blocks of observation table and k is the total number of attributes of observation table.

A. Analysis of storage capacity of EAV

Let n = total number of facts, $q =$ average length of attribute names, $g =$ average length of values. In EAV, 32 bits (4 bytes) is required to represent entity. Size of each fact in EAV is $(4+q+g)$ bytes. Hence, the total size to hold all facts is $S = n \times (4+q+g)$ bytes.

B. Space complexity of medical domain dictionaries and rule base

Let $C_i =$ cardinality of i[th] attribute where domain expert knowledge is not applicable, $L_i =$ average length of i[th] attribute name, *P* = number of categorical attributes. Codes of attributes are not stored explicitly and the index of attribute is the code. Domain dictionary storage of i[th] attribute is $C_i \times L_i$ bytes. Total domain dictionaries storage (SD) is $\sum_{i=1}^{p} (C_i \times L_i)$ bytes. If the size of rule base storage is R, the dictionary and rule base storage (SDR) is $\sum_{i=1}^{p} (C_i \times L_i) + R$ bytes.

C. Analysis of storage capacity of OEAV

Let $p =$ number of attributes, $q =$ average length of attribute names. Total storage of attribute dictionary is $p \times q$ bytes. Let $S =$ size of each block address in byte. Total storage of index table is $p \times q + p \times S$ bytes. In OEAV, 32 bits are required to represent entity and 16 bits are required for attribute and value individually. 64

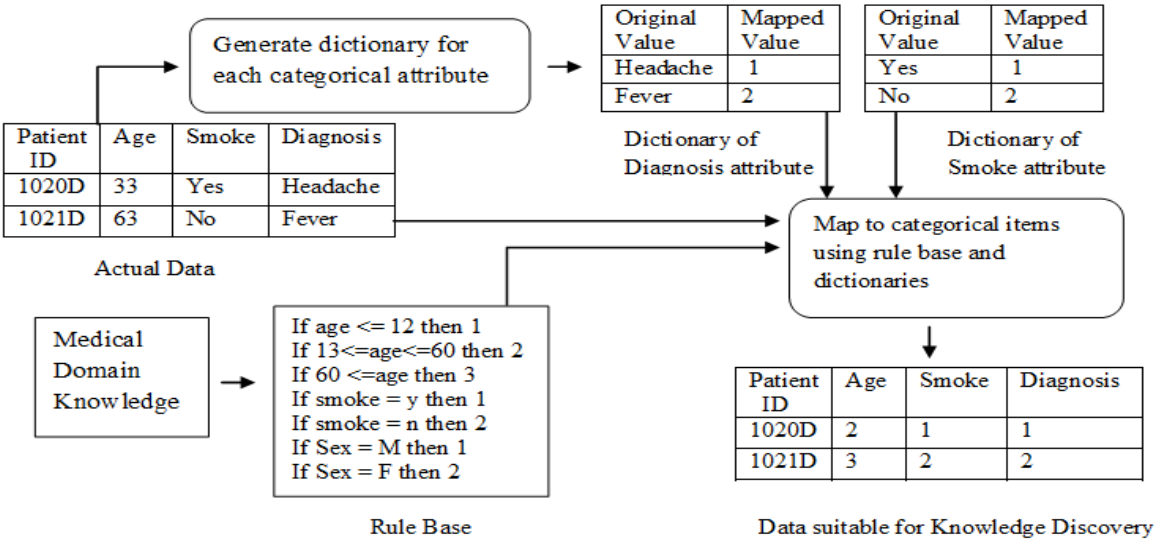


Figure 2. Data Transformation of Medical Data.

bits = 8 bytes = size of each fact in OEAV. Let n = total number of facts, m = total number of facts in a block, w = word size (bytes). Total number of blocks is $\lceil n/m \rceil$. The number of words per fact is $\lceil 64/w \rceil$. For block i where 1≤ i ≤$\lceil n/m \rceil$, the number of words per block is $\lceil (m \times \lceil 64/w \rceil) \rceil$and the size of the block is $w(\lceil (m \times \lceil 64/w \rceil) \rceil)$. Hence the size to hold all facts is $S = \lceil n/m \rceil \times w \times (\lceil (m \times \lceil 64/w \rceil) \rceil)$. In OEAV, total size to hold all facts = storage for facts + storage for domain dictionaries and rule base + storage for attribute dictionary + storage for index table + storage for modified B+ tree $= \lceil n/m \rceil \times w \times (\lceil (m \times \lceil 64/w \rceil) \rceil) + \sum_{i=1}^{p} (C_i \times L_i) + R + (p \times q) + (p \times q + p \times S) + B$ bytes.

Results and Discussion

The experiments were done using PC with core 2 duo processor with a clock rate of 1.8 GHz and 3GB of main memory. The operating system was Microsoft Vista and implementation language was c#. We have designed a data generator that generates all categories of random data: ratio, interval, decimal, integer, percentage etc. This data set is generated with 5000 attributes and (5-10) attributes per transaction on average. We have used highly skewed attributes in all performance evaluations to measure the performance improvement of our proposed open schema data models in worst case. For all performance measurement except storage performance, we have used 1 million transactions.

A. Storage performance

Figure 3 shows the storage space required by EAV and OEAV. The EAV occupies significantly higher amount of storage than OEAV. This is due to the data redundancy of EAV models.

B. Time comparison of projection operations

Figure 4 shows the performance of projection operations on various combinations of attributes. Almost same time is needed with different number of attributes in EAV, as it has to scan all the blocks whatever the number of attributes. In OEAV, it can be observed that the time requirement is proportional to the number of attributes projected. This is because that the query needs to scan more number of blocks as the number of attributes increases.

C. Time comparison of select queries

Figure 5 shows the performance of multiple predicates select queries on various combinations of attributes. Figure 5 shows almost same time is taken with different number of attributes in EAV as it has to scans all the blocks twice whatever the number of attributes in predicate. The graph shows how time is varied in OEAV with different number of attributes as it scans number of attribute partitions proportional to number of attributes in select quires. This experiment shows EAV has taken much higher time compared to OEAV. It is because it has no tracking of how data are stored, so it has to scans all the blocks once to select entities and has to scan all the blocks one more time to retrieve the attribute values for the selected entities. OEAV has taken the lower time as it does not need to read unused attributes to select entities and can retrieve attribute values of these entity without reading any unused attribute value using entity indexing.

D. Time comparison of aggregate operations

Aggregate operations compute a single value by taking a collection of values as input. Figure 6 shows the performance of various aggregate operations on a single attribute. Time is not varied significantly from one aggregate operation to another as different aggregate operations need same number of data block access for most of the cases. Figure 6 shows EAV has taken much higher time than OEAV as it has to scan all the blocks to compute each operation. OEAV has taken negligible time for max, min, count operations on a single attribute as to find max and min it has to scan only 1 block and count result is computed from its index table. For average operation on an attribute, it has taken considerable time, as it has to scan all the blocks of that attribute.

E. Time comparison of statistical operations

Figure 7 shows the performance of various statistical operations on a single attribute. Time is varied significantly from one statistical operation to another as different statistical operations need different sorts of processing. This experiment shows EAV has taken much higher time compared to OEAV. It is because it has no tracking of how data are stored, so it has to scan all the blocks to compute each operation. We can see from this figure OEAV has taken negligible time for median operation as it has to scan 1 or 2 blocks for this operation. For mode and standard deviation, it has to scan all data blocks of the attribute for which particular operation is executing once, twice respectively.

F. Time comparison of CUBE operations

The CUBE operation is the n-dimensional generalization of group-

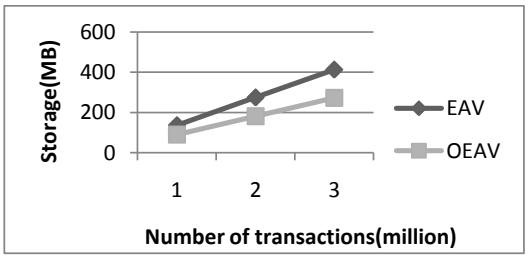


Figure 3. Storage performance.

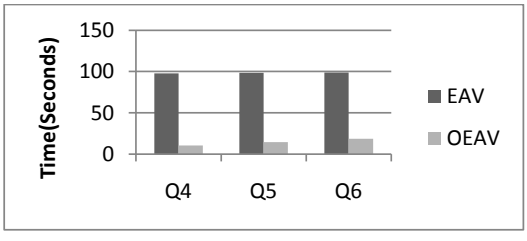


Figure 5. Time comparison of select queries.

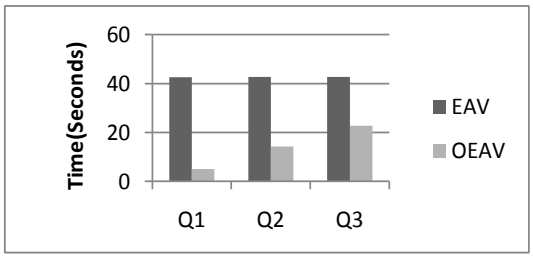


Figure 4. Time comparison of projection operations.

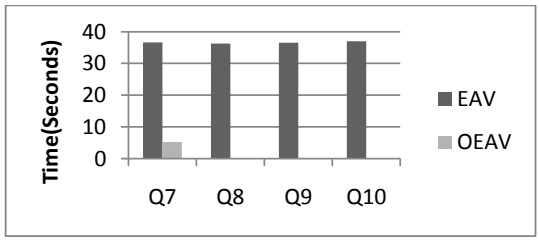


Figure 6. Time comparison of aggregate operations.

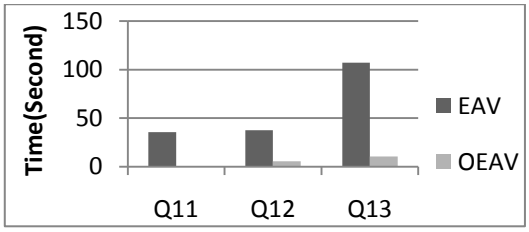


Figure 7. Time comparison of statistical operations.

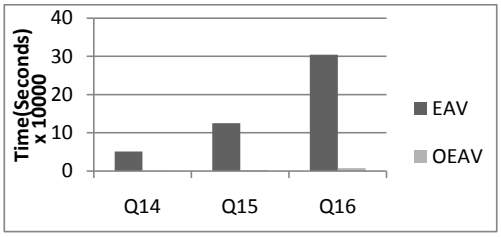


Figure 8. Time comparison of CUBE operations.

Q1: Select A_i from observation;
Q2: Select A_i , A_j ,A_k from observation.
Q3: Select A_i, A_j, A_k, A_l, A_m from observation.
Q4: Select * from observation where A_i='XXX'.
Q5: Select * from observation where A_i='XXX' AND A_j='YYY'.
Q6: Select * from observation where A_i='XXX' AND A_j='YYY' AND A_k='ZZZ'.
Q7: Select AVG (A_i) from observation.
Q8: Select Max (A_i) from observation.
Q9: Select Min (A_i) from observation.
Q10: Select Count (A_i) from observation.
Q11: Select Median (A_i) from observation.
Q12: Select Mode (A_i) from observation.
Q13: Select Standard Deviation (A_i) from observation.
Q14: Select A_i, A_j, Max (Am) from observation CUBE-BY (A_i, A_j)
Q15: Select A_i, A_j, A_k, Max (A_m) from observation CUBE-BY (A_i, A_j, A_k)
Q16: Select A_i, A_j, A_K, A_m, Max (A_n) from observation CUBE-BY (Ai, A_j, A_k, A_n)

by operator. The cube operator unifies several common and popular concepts: aggregates, group by, roll-ups and drill-downs and, cross tabs. Here no pre-computation is done for aggregates at various levels and on various combinations of attributes. Figure 8 shows the performance of CUBE operations on various combinations of attributes. It can be observed that the number of attributes in cube operations leads to the time taken as CUBE operation computes group-bys corresponding to all possible combinations of CUBE attributes. The experiment results show that EAV has taken much higher time compared to OEAV as it does not partition data attributes wise and it has no entity index.

Conclusion

EAV is a widely used solution to model data which are sparse, high dimensional and need frequently schema change, but EAV is not a search efficient data model for knowledge discovery. In this paper, we have proposed a search efficient open schema data models OEAV to model high dimensional and sparse data as an alternative of EAV. We have implemented both EAV and OEAV models in a data warehousing environment and performed different relational and warehouse queries on both the models. We have achieved a query performance faster in the range of 15 to 70 compared to existing EAV model. These efficiencies arise due to binary data representation in OEAV where as string representation is used in EAV model. The experiment results show our proposed open schema data model is dramatically efficient in knowledge discovery operation and occupy less storage compared to widely used EAV model.

References

1. Stead, W.W., Hammond, W.E., and Straube, M.J. (1983). A chartless record--is it adequate? *J Med Syst* 7, 103-109.
2. Thomas, E.J., Jeffrey, T.W., and Dubbels Joel, C. (2007). A health-care data model based on the HL7 reference information model. *IBM Systems Journal* 46, 5-18.
3. Li, J.L., Li, M.X., Deng, H.Y., Duffy, P.E., and Deng, H.W. (2005). PhD: a web database application for phenotype data management. *Bioinformatics* 21, 3443-3444.
4. Anhoj, J. (2003). Generic design of Web-based clinical databases. *J Med Internet Res* 5, e27.
5. Brandt, C.A., Deshpande, A.M., Lu, C., Ananth, G., Sun, K., Gadagkar, R., Morse, R., Rodriguez, C., Miller, P.L., and Nadkarni, P.M. (2003). TrialDB: A web-based Clinical Study Data Management System. *AMIA Annu Symp Proc*, 794.
6. Nadkarni, P.M., Brandt, C., Frawley, S., Sayward, F.G., Einbinder, R., Zelterman, D., Schacter, L., and Miller, P.L. (1998). Managing attribute--value clinical trials data using the ACT/DB client-server database system. *J Am Med Inform Assoc* 5, 139-151.
7. Nadkarni, P. http://ycmi.med.yale.edu/nadkarni/Introduction%20-to%20EAV%20systems.htm. Yale University School of Medicine. [Online].
8. Pin-Shan Peter, C. (1976). The entity-relationship model-toward a unified view of data. *ACM Transactions on Database Systems* 1, 9-36.
9. Hoque., A.S.M.L. (2002). Storage and querying of high dimensional sparsely populated data in compressed representation. *Lecture Notes on Computer Science*, 2510, 418-425.
10. Agarwal, S., Agrawal, R., Deshpande, P., Gupta, A., Naughton, J.F., Ramakrishnan, R., and Sarawagi, S. (1996). On the Computation of Multidimensional Aggregates. *In Very Large Data Bases* 506-521.
11. Adam, B., Jim, Gray., Andrew, Layman., Hamid, Pirahesh. (1997). Data cube: a relational aggregation operator generalizing group-by, cross-tab and sub-totals. *Data Mining and Knowledge Discovery* 1, 29-53.